EN.601.270
Open Source Software Engineering

Stephen R. Walli

Welcome!

- Course Objectives
- Marking & Grades
- Textbooks, Notes, Office Hours, etc.
- A couple of demonstrations to set the stage …
- Student projects kick-off

My Objectives (Why I'm Here …)

- There is a scarcity of software engineering skills in the industry
- There is a scarcity of knowledge about open source software
- These scarcities lead to bad outcomes for solutions, products and services

Course Objectives

- Understand software engineering and the difference between programming-in-the-small, and the tooling and process to support software engineering at scale
- Understand how liberally licensed, collaboratively developed software fits into the world of a professional programmer
- Gain real-world experience in programming-in-the-large in an open source software project community
- Be able to identify healthy open source projects and tools you can use in your software projects without compromising your work
- Be aware of the high-level community and business issues (ownership, licenses, non-profits, community development, culture)
- Understand the basic framework for intellectual property that every professional developer needs to know

Textbooks, Notes, Office Hours, etc.

- All the class notes, papers, etc. are on GitHub https://github.com/jhu-ospo-courses/JHU-EN.601.270
- Blackboard will be used for announcements, forums, grading, testing
- Text Books (PDF available on Blackboard):
  - *Producing Open Source Software*, Karl Fogel (2020)
  - Intellectual Property and Open Source, Van Lindberg (2008)
- Past papers for class discussion on the Course GitHub site
- Office Hours:
  - Weds, 12:00-14:00, Malone 216
  - Weds, 16:00-18:00, Zoom: 465 974 4379

Marking and Grades

- Term student open source project 50% of the final grade
  - Mid-term mark, **18 October** = 25%
  - 2nd Half-term mark = 25%
- Two term tests
  - Test #1, **28 September** = 20%
  - Test #2, **2 November** = 20%
- Engagement = 10%
  - Use the class forum discussions around the weekly readings

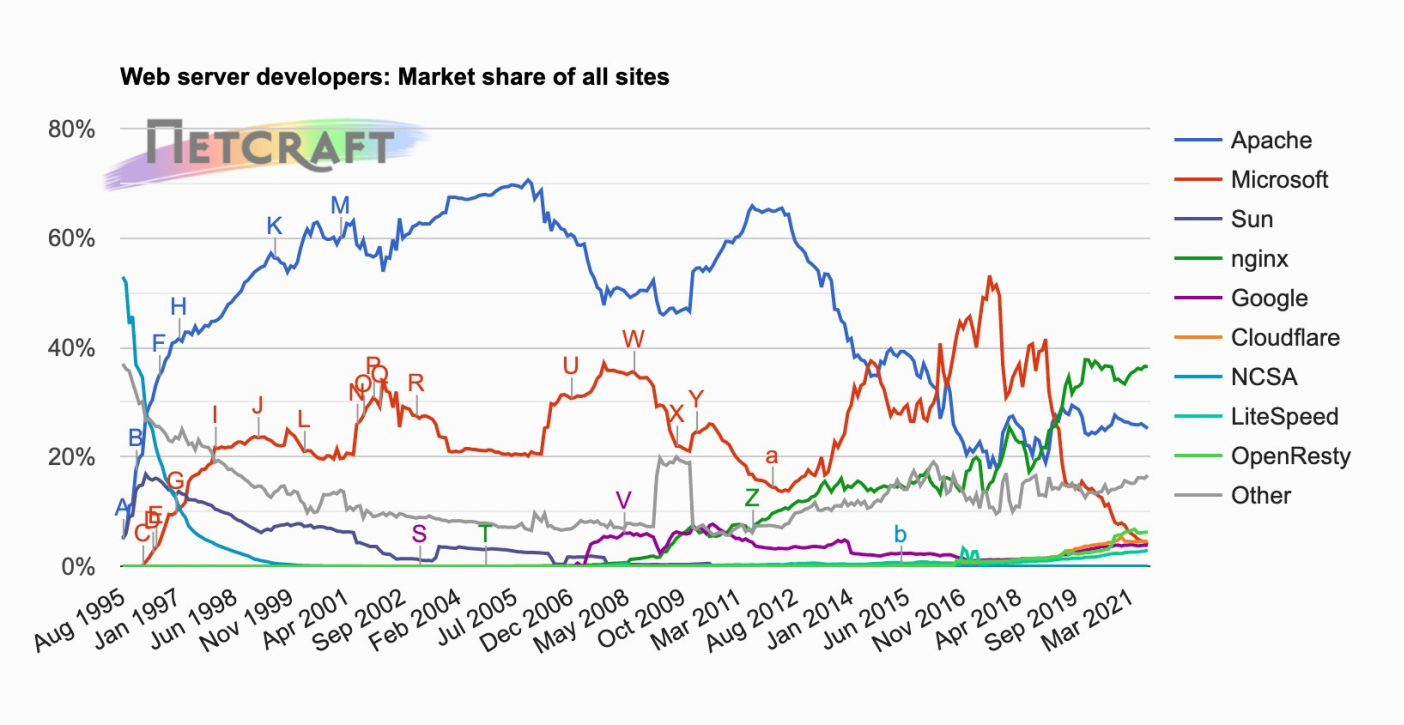
Syllabus

- Engineering Economics of Open Source
- Healthy Open Source Software Project Practices
- Topics in Collaborative Engineering (Community Development, Non-profits, Culture)
- Topics in Software Engineering (Life-cycle, Construction & Testing, Deployment & Distribution, Operations & Support)
- Intellectual Property for Engineers
- Two ”Ask Me Anything” sessions

Open Source Software

Let’s Build Apache httpd

- This is the classic webserver
- By 2000 roughly 60% - 70% of the worlds web traffic ran this way (now ~25%)
- Following the Lab 1 instructions from Intersession Course (JHU-EN.601.210)



https://news.netcraft.com/archives/category/web-server-survey/
https://github.com/jhu-ospo-courses/JHU-EN.601.210/tree/main/labs/1

So what …

COCOMO RESULTS for Apache								
MODE	**"A" variable**	**"B" variable**	**"C" variable**	**"D" variable**	**KLOC**	**EFFORT, (in person-months)**	**DURATION, (in months)**	**STAFFING, (recommended)**
semi-detached	3	1.12	2.5	0.35	280.000	1651.727	33.436	49.399

Explanation: The coefficients are set according to the project mode selected on the previous page, (as per Boehm). Note: the decimal separator is a period.

The final estimates are determined in the following manner:

effort = a*KLOC^{b}, in person-months, with KLOC = lines of code, (in thousands), and:

staffing = effort/duration

where a has been adjusted by the factors:

280K LoC (refactored)
1650 months == 137.5 years
At $100K/year * 1.5 (gross up) == $150K
$20.6M of software value

https://strs.grc.nasa.gov/repository/forms/cocomo-calculation/
https://www.indeed.com/career/software-engineer/salaries

But

- Apache httpd has been refactered – this build was just the base
- Linux (~50M LoC)
- Docker
- gcc
- configure (autoconf/automake), make
- Perl
- All of this is open source licensed and built in collaborative communities

Collaboratively Developed Liberally Licensed

Software Engineering

Round 1 (A Very Low-Tech Analog Demo)

How it works

- Each envelop is a release of an application
- The card inside:
  - Side A – Deliverables in the release in brackets e.g. [App] [Docs]
  - Side B – User comments in {braces}, Developer comments in (parentheses)
- There might be a second sometimes …

Round 1

Comments:

I can’t install it on my X machine (or my phone)

The install failed saying I don’t have a current N-language runtime

This app is AWESOME and I wish it also did FIZZLE

This app is REALLY AWESOME.

This app would be better if it also did FOO

This app would be better if it also did XYZ

The APP seems to get the time wrong

If you give me the source code, I could add a BAR function

Round 1

Let’s talk about:

1. Bucketing responses.
   - ISSUES & BUGS
   - FEATURES
   - JOY
2. Do we need [install] docs?

Round 2

Round 2

Application Comments:

I still can’t install it on my Y machine.

This app is AWESOME.

This app is (still) REALLY AWESOME.

This app would be better if it also did FOO

This app would be better if it also did ABC

If you give me the source code, I could add a SQL function

Developer Comments:

I don’t know how to build App

I don’t know if I correctly built App

Round 2

Ideas:

1. Better bug analysis and reporting
2. Better user docs
3. Build instructions
4. Testing instructions
5. Licensing and that patch

Round 3

Round 3

Application Comments:

This app is AWESOME.

This app is (still) REALLY AWESOME.

This app would be better if it also did FOO

This app would be better if it also did FOOZLE

This app would be better if it also did SQL2

The app destroyed my App Data

Round 3

Application Comments:

This app is AWESOME.

This app is (still) REALLY AWESOME.

This app would be better if it also did FOO

This app would be better if it also did FOOZLE

This app would be better if it also did SQL2

The app destroyed my earlier App Data [this user didn't install version 2 but went from version 1 to 3!]

Round 3

Ideas:

1. Branching
2. Install testing … really User eXperience testing
3. Versions and semantic versioning
4. Iterative improvements in all the things …

$$1 \rightarrow 10 \rightarrow 100 \rightarrow 1000 \rightarrow \ldots \rightarrow 1000000$$

Computer Science vs. Software Engineering

$1 \rightarrow 10 \rightarrow 100 \rightarrow 1000 \rightarrow \ldots \rightarrow 1000000$

Software Engineering is delivering software at scale

Scaling for users, developers, complexity

Student Open Source Term Projects

Student Projects

- A number of open source projects have developed student projects for you as real-world working experience
- Each open source project has a number of mentors that will help you accomplish your student project through the semester
- You need to choose a student project: **It is first come, first served.**

Choosing Your Student Project

- YOU need to send ME email (swalli1@jh.edu)
- Tell me the 3 or 4 projects in order of preference that you would like and I will make best efforts to assign you to your higher preferences
- Name your preferences by the OSS Project + Number + Title, e.g.,

  OpenCRAVAT #2: Gene set analysis
  PASS #2: Develop a PASS dashboard for super users
  Microsoft Powershell #8: Add tab completion for Scope parameters
- I will email introduce you to your project mentor and your project mentor will check with you that you understand the project and its desired outcomes
- DON'T disqualify yourself – talk to the mentor!
- This matching needs to be finished quickly

Choosing Your Student Project - II

- IF you really really want to work in the same open source group of student projects with a classmate:
  - LET ME KNOW IN YOUR EMAIL TO ME (naming your classmate, and copying them on the email)
  - YOU EACH STILL NEED TO EMAIL ME
  - We will make best attempts to accommodate this desire

The Messiness of the Real World

- The mentors followed a template for student projects, BUT they are also all different in slightly different ways – this is the real world
- Student projects were estimated at 50-60 hours of work
- There is an estimated difficulty – sometimes very broadly defined – and that means the end result may be broadly defined
- You want to be making progress each week, working with your mentor
- **Remember there is a mid-term grade on work done to date**

The Mentors

- Marks come from me based on my discussions with your mentors
- Weekly mentor checkins, i.e. the equivalent of “mentor office hours”
- Work with your mentors:
  - To get up-and-running to a built state
  - To plan an approach to the work so you can be making weekly progress
  - To understand any additional reading you might need to do
- If you have trouble reaching your mentor – reach for me quickly by email

Course Site(s)

Class Notes:

https://github.com/jhu-ospo-courses/JHU-EN.601.270

Student Projects:

https://github.com/jhu-ospo-courses/JHU-EN.601.270/tree/main/Student%20Projects

Blackboard for announcements, videos, textbook PDFs, etc.